WORK BALANCE AGREEMENT:

- ❖ Producing deliverable
- ❖ Testing/reviewing deliverable
- ❖ Dates by which test, deliverable, or review

Submittables:

- ☐ Class diagrams
- ☐ Interaction Diagrams
- ☐ Design Rationale

Class Breakdown Structure (A2):

REQUIREMENT:	OVERVIEW:	PERCENT:	PERSON:
Requirement 1	Player and Estus Flask	8 (ALL)	Nic
Requirement 2	Bonfire	4 (player, flask, enemies, weapons, skills)	Nic
Requirement 3	Souls	3 (player, enemies, vendor)	Ting
Requirement 4	Enemies	8 (player, souls, skills, weapons, terrain, shrine)	Ting
Requirement 5	Terrains	3 (player, enemies)	Ting
Requirement 6	Death (dying in game)	8 (ALL)	Isha
Requirement 7	Weapons	4 (player, skills, enemies, vendor))	Isha
Requirement 8	Vendor	4 (player, souls, weapons, shrine)	Nic

Nic: I accept this WBA.

Isha: I accept this WBA.

Ting: I accept this WBA.

Assignment 1 Member Tasks:

NIC	ISHA	TING
☐ Review Final Design Rationale ☐ Review Final Sequence Diagram	☑ ~~Brush Up UML~~ ☐ Complete Sequence Diagram ☐ Review Final UML class diagram ☐ Review Final Design Rationale	☑ ~~Brush Up UML~~ ☐ Complete Design Rationale ☐ Review Final UML class diagram ☐ Review Final Sequence Diagram

Work Breakdown Structure (A1):

	Isha	Nic	Ting	Running Percent:
UML Diagrams		Drafted		90%
Interaction Diagrams	IN PROGRESS			
Rationale			IN PROGRESS	
Revised UML	Complete	IN PROGRESS	Complete	95%
Revised Interactions				
Revised Rationale				
Meeting 1:	Complete	Complete	Complete	100%
Meeting 2:	Complete	Complete	Complete	100%
Meeting 3:				
			Total:	50%

Game Notes:

NIC	ISHA	TING
PLAYER & ESTUS: ➔ Use me :3 ***BONFIRE:*** ➔ Use me :3 ***VENDOR:*** ➔ Use me :3	***DEATH:*** ➔ When player dies -> print message ➔ RESET features executed ➔ Respawn to most recent bonfire ➔ Soul level becomes 0 -> Player can interact with token to regain lost souls ➔ Dying from falling -> Token appears one step before dying location ➔ Player dies before retrieving token of souls -> Token replaced with latest one ***WEAPONS:*** ➔ Only able to equip one weapon at a time ➔ Cannot drop weapons intentionally/when dying ➔ Weapon has active/passive skills ➔ Enemies can use weapon skills randomly ➔ May use any display character for each weapon	***SOULS:*** ➔ Souls ⇒ Money. ➔ Default: 0 Souls. ➔ Broadsword = 500 Souls. ➔ Giant Axe = 1000 Souls. ➔ Storm Ruler = 2000 Souls. ➔ Undead = 50 Souls. ➔ Skeleton = 250 Souls. ➔ Yhorm = 5000 Souls. ➔ Can be traded or buy new weapons or upgrades from vendor. ***ENEMIES:*** ***TERRAINS:***

1 player has 1 weapon
Estus flask has three recharges?
Hit points for different enemies?
4-12 skeletons in game
How many souls each enemy gives
1 boss (Yorm)

- BOTH UML and INTERACTION ON SAME PDF

Design Rationale:

The design of the Souls Program can be improved by the following principles:

1. **Don't Repeat Yourself principle:** In which the system will have a piece of logic with a specified and unique function. This can be represented by utilising abstract classes, interfaces and public constants.

Abstract classes: To provide a common function across a set of related classes while also allowing default class implementations. This can be represented by having an "enemies" class where the undead, lord of cinder and skeleton have a common functionality which is attacking the player. This class will then be the parent class and is inherited by child classes like undead, skeleton and lord of cinder classes. On the other hand, the same logic can be applied to the terrain functions. The terrains determine the location of the enemies and the player's safe space which is within the walls. Having one common function which is limiting the location of the enemies within the walls and the player entering the valley.

- Terrains inherit from Ground
- Actions methods inherit from action

2. **Liskov Substitution Principle (LSP):** Where a derived class can be used as a substitute without changing the behaviour of the main class.
- Terrains inheriting from enemy location classes without

New classes:

- Skeletons
- Souls
- Token of Souls
- Vendor
- Bonfire
- Cemetery
- Resettable